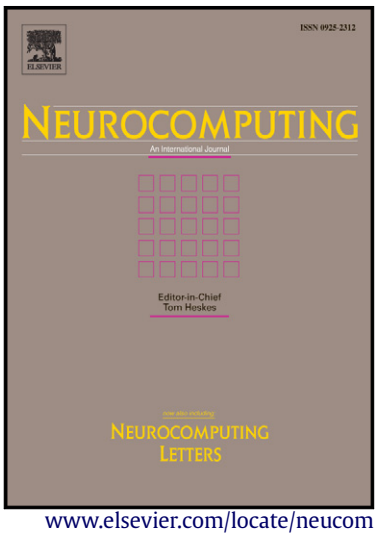

www.elsevier.com/locate/neucom

Implementation of context-aware workflows with Multi-agent Systems

Javier Alfonso-Cendón, José M. Fernández-de-Alba, Rubén Fuentes-Fernández, Juan Pavón

PII: S0925-2312(15)00546-9
DOI: http://dx.doi.org/10.1016/j.neucom.2014.10.098
Reference: NEUCOM15442

To appear in: *Neurocomputing*

Received date: 10 March 2014
Revised date: 9 October 2014
Accepted date: 21 October 2014

Cite this article as: Javier Alfonso-Cendón, José M. Fernández-de-Alba, Rubén Fuentes-Fernández, Juan Pavón, Implementation of context-aware workflows with Multi-agent Systems, *Neurocomputing,* http://dx.doi.org/10.1016/j.neucom.2014.10.098

Implementation of Context-Aware Workflows with Multi-Agent Systems

Javier Alfonso-Cendón[1], José M. Fernández-de-Alba[2], Rubén Fuentes-Fernández[2], and Juan Pavón[2]
[1] Escuela de Ingenierías Industrial e Informática, Universidad de León
24071 León, Spain
javier.alfonso@unileon.es
[2]Facultad de Informática, Universidad Complutense de Madrid
Avda. Complutense, s/n. 28040 Madrid (Spain)
{jmfernandezdealba,ruben,jpavon}@fdi.ucm.es

Abstract. Systems in Ambient Intelligence (AmI) need to manage workflows that represent users' activities. These workflows can be quite complex, as they may involve multiple participants, both physical and computational, playing different roles. Their execution implies monitoring the development of the activities in the environment, and taking the necessary actions for them and the workflow to reach a certain end. The context-aware approach supports the development of these applications to cope with event processing and regarding information issues. Modeling the actors in these context-aware workflows, where complex decisions and interactions must be considered, can be achieved with multi-agent systems. Agents are autonomous entities with sophisticated and flexible behaviors, which are able to adapt to complex and evolving environments, and to collaborate to reach common goals. This work presents architectural patterns to integrate agents on top of an existing context-aware architecture. This allows an additional abstraction layer on top of context-aware systems, where knowledge management is performed by agents. This approach improves the flexibility of AmI systems and facilitates their design. A case study on guiding users in buildings to their meetings illustrates this approach.

Keywords: Context-aware systems; Workflow design; Multi-Agent Systems; Ambient Intelligence.

1. Introduction

Ambient Intelligence (AmI) systems integrate a diverse set of technologies and devices in order to provide services to users in an unobtrusive way. Raw data coming from sensors of the environment are aggregated and filtered to create more abstract information [22], which can be processed by application components at a higher level [20], in order to decide what actions should be performed on the environment [21].

This process involves several activities. For instance, finding the available sources of information and their types, gathering the data from these sources, facilitating the fusion (aggregation and derivation) of the different pieces of data [4][19], building and updating a representation of the environment with this information (what is known as the *context*) to be used by applications, and triggering actions in actuator devices [8]. This kind of environment is highly dynamic because of the changing behavior of human users, as well as rearrangements in system topology when devices fail, or are added or removed.

These activities and the adaptation to changing conditions can be organized in terms of *workflows*. These workflows are context-aware because their activities are triggered by changes in the *context*, a representation of the state of the environment. They may involve multiple actors, including systems and users, which require coordination. Adaptation here implies performing tasks according to the actual context taking into account, for instance, resources and user configuration. A correct evaluation of the context relies on systems making a proper interpretation of the available data, and using the inferred context to fill in the missing information needed by their services. This adaptation requires an infrastructure that resolves abstract representations of existing tasks into runtime processes that operate the sensors and actuators of the smart environment, either to monitor or to perform the required actions. It also needs to integrate the domain logics of different entities, both human and physical devices (see, for instance, the management of the context with mobile devices in [6]).

A suitable way to design such information workflows is by applying the concept of Multi-Agent Systems (MAS). MAS allow modeling distributed heterogeneous entities (the agents) and their cooperation (for instance, in a workflow) towards the accomplishment of common goals. The application of MAS for designing context-aware workflows in this case can take advantage of other works that have validated the use of MAS for AmI. There are works approaching this integration from general and methodological perspectives focused on system development [27, 29] and knowledge management [25]. Other works address specific aspects of the problem, such as sensor integration [2], the interpretation of gathered data [12], or the use of reputation techniques to evaluate the trustability of service providers and manage user' profiles and their related security issues [24].

Although MAS have been used for processing context-aware information in AmI applications [27], many of them are too focused on low abstraction levels or domain-specific tasks. For instance, agents are integrated at hardware level in [1, 28]. In order to get a more systematic way to apply MAS in AmI applications, it would be convenient to be able to work with context representations at different levels of abstraction. This requires processing raw data to extract relevant information to

dynamically build the context at a higher level of abstraction. This facilitates the semantic processing of information by agents in order to make consistent decisions on how to act on the environment. The organization of this process is what is described in this paper as *context-aware workflow*, because it is triggered by changes in the context and finally acts on the environment, with the corresponding effects on the context.

To put all pieces together, this paper presents an architecture where MAS manage context-aware workflows for AmI applications. The design of the MAS uses common agent design concepts, as extracted from the INGENIAS agent-oriented methodology [13]. The infrastructure for developing the AmI system is FAERIE (Framework for AmI: Extensible Resources for Intelligent Environments) [11], which supports distributed management of AmI contexts and workflows. This framework can be downloaded from [14]. FAERIE proposes splitting the context representation into several abstraction layers. Each layer considers certain information and its processing, both horizontal (i.e., on the same abstraction layer) and vertical (i.e., between successive abstraction layers). It also establishes how to coordinate the components of these layers in order to process information, which facilitates reasoning on the context. Workflow management (i.e., detection, tracking and execution of workflows) is built upon the previous functionality. Workflows are represented as activity diagrams that operate by using abstract expressions as data. The context aware applications can make use of all these features.

This approach is illustrated with a case study of a system that guides a user in a building to meet a person. The system has information about a map of the environment (in this case, rooms, corridors, stairs, etc., in the building), and an activity diagram that describes the guiding process (i.e., the workflow). The context information changes to show the location of people in the building. The workflow status is updated to track users and to deduce the completion degree of the workflow. The infrastructure coordinates the available sensors and actuators to perform the actions described in the workflow. A software agent is the actor responsible of the workflow for the guiding activity. This agent uses dialog management to interact with the user.

The rest of the paper is organized as follows. Section 2 reviews alternative approaches to deal with context and workflow management in context-aware systems and their tradeoffs. Section 3 presents the support developed in FAERIE for context-aware workflows, and Section 4 the software architecture for the integration of MAS with them. The case study in Section 5 illustrates its use with an application to guide a person in a building, and shows a complete execution of the workflow. Section 6 uses these results to discuss the evaluation of the approach. Finally, Section 7 presents some conclusions and proposals for the evolution of the approach.

2. Management of context-aware workflows

Most approaches for the management of context-aware workflows clearly separate the control of the workflow execution from the management of context information. The works of Ranganathan and McFaddin [23], uFlow [16] and CAWE [3] are examples of this. The first two propose wrapping the context management system and using it to check conditions in a workflow execution environment, while the third wraps the workflow management as another context provider/consumer into a context aware architecture. The approach of Ranganathan and McFaddin uses the context-aware component in order to choose a suitable workflow definition, and then proceeds to its execution. However, it does not consider the changes in the conditions affecting that choice once it has been done. On the contrary, the other two approaches work with an abstract workflow definition, in which actions are instantiated at runtime depending on current context conditions. The approach chosen in FAERIE [11] is similar to that of uFlow, but it supports the use of any workflow definition language, as the corresponding engine is wrapped with modules that adapt the inputs and outputs as required. This is not the case for uFlow and CAWE, which describe definition languages of their own, and of Ranganathan and McFaddin, which propose using BPEL (Business Process Execution Language), a standard language for the definition of business processes. The advantage of the first two alternatives over the third is that they include context dependent concepts. This allows defining workflows conditions and actions in terms of context information. However, the third alternative uses a well-known and established language, which facilitates its reuse in different contexts.

The use of the agent paradigm is also widespread in AmI literature [29]. For this work, it is of particular interest their relationships with the management of context information and workflows.

Regarding the management of context information, works are usually focused on solving specific tasks taking advantage of the knowledge-based capabilities of agents. For instance, the extension of the MAS Amadeus in [15] provides capabilities to learn the user's behavior. Some more general works try to provide formal descriptions of knowledge for some tasks and procedures to apply it. This is the case of [17], which provides an ontology for intrusion detection events, and uses prevention rules based on pattern detection and clustering algorithms to work on it. Finally, works like [26] provide full architectures for context management. Their main issue is their commitment to quite specific structures for the resulting systems. This is useful to facilitate development guidelines and understand the running systems, as it adjusts to

a known model. However, this reduces flexibility as it imposes a structure that may not be well-suited for the problem at hand.

Illustrative examples of the use of agents to manage workflows in AmI systems are the projects of Aiello et al. [1, 2], [7], and CAKE [5]. The work in [1] describes a framework to program light agents for Wireless Sensor Networks (WSN), while [2] in focused on mobile agents on this domain. They do not provide support for defining and monitoring workflows, but [2] considers the detection of activities by means of body sensors. The definition of these activities is mainly achieved through assisted automatic learning, as there is no explicit definition of them. This is the same approach adopted in [7], in this case to determine activities from accelerometer data. The automated discovering and learning of activities offers great flexibility for monitoring activities and changing their definition in runtime. The drawback is the usual limited complexity of the learned activities regarding to types of actions and number of participants. The other example shown here is CAKE [5]. Its agents organize themselves using abstract workflow definitions, depending on the situation details obtained from a case-based reasoner. However, the architecture does not offer specific support to facilitate information interpretation in order to identify a suitable case. It leaves to the agent implementations the task of determining which the situation is to request the case. Anyway, this kind of work is more concerned with providing infrastructures, libraries, or procedures to develop MAS using them, than with how to use these MAS to provide context-aware services.

The previous references highlight several requirements and problems in the use of context-aware workflows and their integration with agents. First, there is a need to decouple the low-level aspects of context management, which are the responsibility of the context-aware platform, from the management of workflow issues. In both cases, there are not predominant approaches in the AmI community. Therefore, the solutions need to provide flexible means to encapsulate these aspects. Regarding the role of agents, there is an interest in using then for knowledge-based aspects and complex interactions. However, most of approaches have focused on infrastructure-level issues or adopted too specific solutions.

3. Context-Aware Workflows in FAERIE

The architecture of the FAERIE framework is based on distributed blackboard models [9]. A system is conceived as a set of interconnected *environments*, each one with its own blackboard (i.e., the *context container*) and a set of components working on it (i.e., the *context observers*). When a given context observer wants to perform a task that needs some information from the context, it makes a request to the context container and waits for its response. Upon this request, the context container publishes the request for information and notifies the other context observers. If any

of these observers can provide the required information directly, then, the request is answered. If not, some context observers may need to make successive requests for information, including requests to remote environments. In this way, the overall system is a federation of environments that supports component collaboration through shared information.

The functionality of the FAERIE architecture is divided into layers. The *Component Infrastructure* layer integrates services from standard component-based frameworks, including component lifecycle management, and the dynamic discovery and binding of abstract services. The *Context Management* layer offers mechanisms to facilitate access to and manipulation of context. Its main services are mechanisms to request changes in the context and subscribe to events that inform on these changes. The framework handles the creation of information flows among the different components that are responsible for interpreting and transforming the knowledge to different levels of abstraction. An additional mechanisms that this layer provides is a way to declare behavior changes depending on context conditions, by making use of the subscribe service to check them [10]. Finally, the *Workflow Management* layer includes the patterns to manage the status of an activity represented as a workflow, which will be explained below. The final and topmost layer, which does not belong to FAERIE, is where context-aware applications are built using the services of the previous layers.

The pattern for workflow management in FAERIE takes as input a workflow definition, i.e., a set of interconnected activities described in some language. From it, the pattern creates four modules. Figure 1 shows this instantiation of the pattern, being *a1*, *a2*, and *a3* the activities of a concrete workflow.

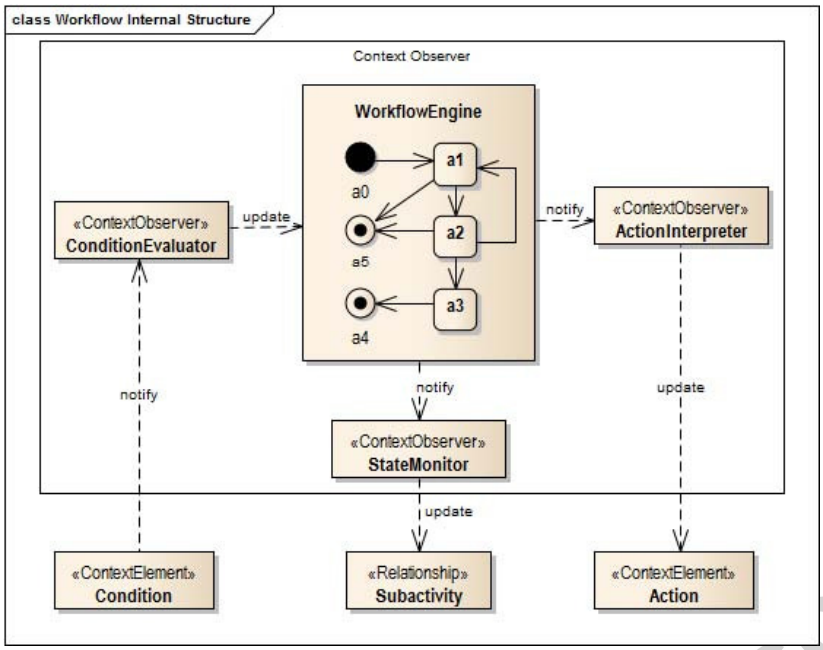

Figure 1. Workflow Pattern Structure.

The modules are:

- *Condition Evaluator*. The evaluator component is in charge of intermediating between the activities requesting information and the context. It publishes these requests in the context container, triggering bottom-up evaluations from sensors to data consumers. The results are used in the *workflow engine* to solve decision nodes or to generate signals. The evaluator works as an adapter that removes context management from workflow definitions.
- *Workflow Engine*. This component stores the abstract definition of the workflow and manages its current state. It receives information from the *condition evaluator* to determine whether a particular activity has been completed; or in order to select the appropriate transitions at decision nodes. It also determines the output to send to the *action interpreter* according to the task to be performed.
- *Action Interpreter*. It receives abstract actions from the *workflow engine*. These actions indicate changes in the context, so they can be redirected to other components of the system. The interpreter asks for these changes, and the context container notifies them to other components. This triggers a top-down evaluation that ends in components that manage actuators.
- *State Monitor*. These workflows can also be seen as context entities whose current state may result interesting for some components. This component monitors the state of a workflow by asking the workflow engine. It publishes this information in the context container if requested.

This structure promotes decoupling between the vocabulary of the domain for the context and the one of the logic of the workflows, fostering their reuse. It also provides ways of implementing the actions and evaluation of conditions as context sensitive elements. Finally, this pattern is the basis for supporting nested workflows (an activity whose implementation is another workflow) and instantiation of workflows (an activity that creates a request from another independent workflow). These elements allow the definition of abstract workflows at different levels of abstraction and their instantiation according to the actual participants.

4. MAS Integration with Context-Aware Workflows

The workflow pattern of FAERIE is the integration point for MAS. There are two alternative ways of performing this integration. The first one is simpler, as it only has to do with the agent external interfaces, but also limits the benefits the agent can get of using the FAERIE infrastructure. The second one may require changes in the internal agent architecture, but also simplifies its development and provides a tighter integration with the underlying platform.

In the simplest case, different implementations for agents may be used as long as they provide a *context observer* interface. This makes them able to observe and change context information, and to define different behaviors depending on context conditions. In this way, agents benefit from the capabilities of the framework to deal with changing resources, so their design may be undertaken in a more abstract way.

The second approach needs that individual agents possess their own “mental state”, as in many agent theories. This mental state acts for the agent as a private context container where it manages its own information. The proposed architecture integrates this mental state with the management of the rest of the context in the *context containers*, as shown in Figure 2.

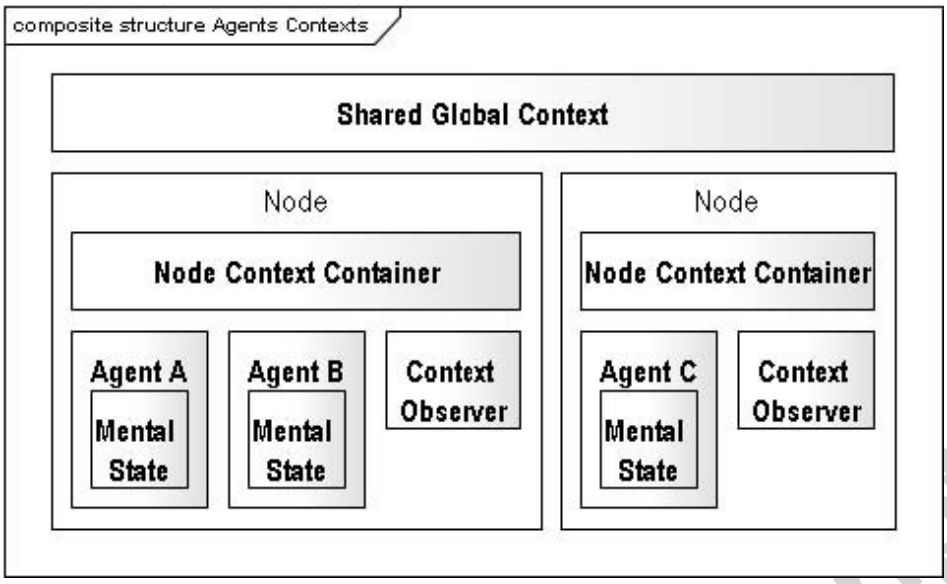

Figure 2. Structure of the Context Container in a Distributed System with Agent Integration.

The *mental state container* acts as the previously described context container. Each time an agent creates an entity to update the mental representation of its own state and that of the environment, it is able to specify whether the entity is created publicly or privately. When it does this, the agent checks whether it is capable of resolving the request by itself, so it can be private information; if not, the request will be forwarded to the context container of its node, becoming publicly available in it. This creates three levels of access in context:

- *Agent Context*. It includes those elements that are only used for the agent's internal reasoning.
- *Node Context*. For those context elements needed to carry out the functionality of the local FAERIE environment that needs contributions from different components.
- *Context-aware systems*. This context comprehends the elements that are shared among different FAERIE environments. This container does not exist physically, as it is the result of the actions of the information-sharing mechanisms among nodes.

This mechanism works in the same way as when the local context container asks a remote environment for information. It makes possible to prevent the node context container from storing information that is not of interest or publicly available for other components of the node. Moreover, the existence of a context container accessible by different agents reduces their exchange of messages to "inform" the other agents, since they have access to most of the basic information on the current environment.

The design and mental state of agents are represented using common concepts from the agent paradigm. Figure 3 in next section shows an example from the case study following the INGENIAS notation [13]. The definition of the *GuideAgent* considers the *goals* it pursues, together with the *tasks* it can perform and the *workflows* in which it participates in order to achieve those *goals*. *Tasks* produce and consume pieces of information (e.g., *events* and *frame facts*), and use environment elements (e.g., *resources* and *applications*).

Different agent-oriented infrastructures support working with the previous kinds of abstractions. In the case of INGENIAS [13], this work is organized around the INGENIAS Development Kit (IDK). It is a modeling environment with code generation functionality. Although it can be used with different target platforms, the INGENIAS Agent Framework (IAF) has been specifically developed to facilitate the alignment between code and INGENIAS specifications. The IAF provides the basic libraries for the agent coding and debugging tools.

Exploiting the abstraction level and infrastructure provided by the agent paradigm facilitates the development of complex AmI scenarios, which may involve multi-agent interactions, automated learning, planning, and decision making. In this way, basic FAERIE context observers can manage the execution of simple tasks, while the most complex ones will be defined by means of MAS.

5. Case Study: The Meeting Guider

To illustrate the previous patterns, this case study considers a system that supports a flexible tuition system. A school wants to arrange rooms for tuition dynamically, according to the actual number of people attending and the resources required, or the room where the teacher is (e.g., a laboratory or an ongoing tuition class). This schema needs students being able to find the room where a tuition will take place even if this has changed lately. The part of the system considered in this case study is responsible for finding out the location of students teachers, and rooms, and guiding the first two groups across the building to the room. The system uses spoken information for guidance, which is interactive and opportunistic. This means that the indications are activated only if the system considers it necessary or the user requests it via voice. This may produce a dialog between the system and the user. The tracking uses the user's location to monitor that certain activities have been finished.

The element responsible for handling the guidance is defined as a software agent (i.e., the *GuideAgent*). The kind of proposed autonomous and reflexive behavior can be modeled and developed in terms of agent theory, i.e., using elements such as goals, information, protocols, and interactions. Figure 3 summarizes it using the INGENIAS [13] notation. The work of this agent takes as basis the formal definition of the

workflow and the infrastructure provided by FAERIE. Using this, the agent is able to read the current state of the workflow, along with the necessary information from the context, in order to establish a dialog with the user. This dialog is a complex task that needs to take into consideration specific knowledge, such as dialog management and natural language understanding. Its implementation using agents has been studied in the literature [18].

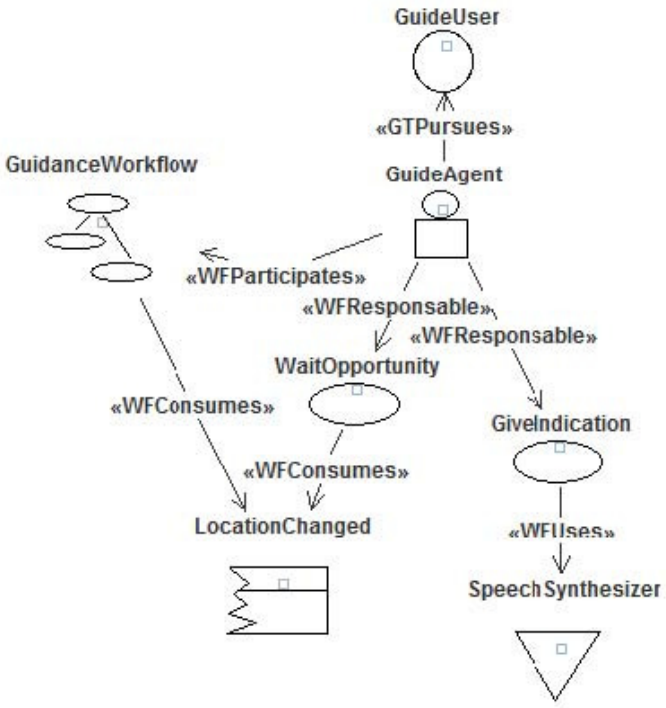

Figure 3. Model of Goal and Tasks for an Example Agent.

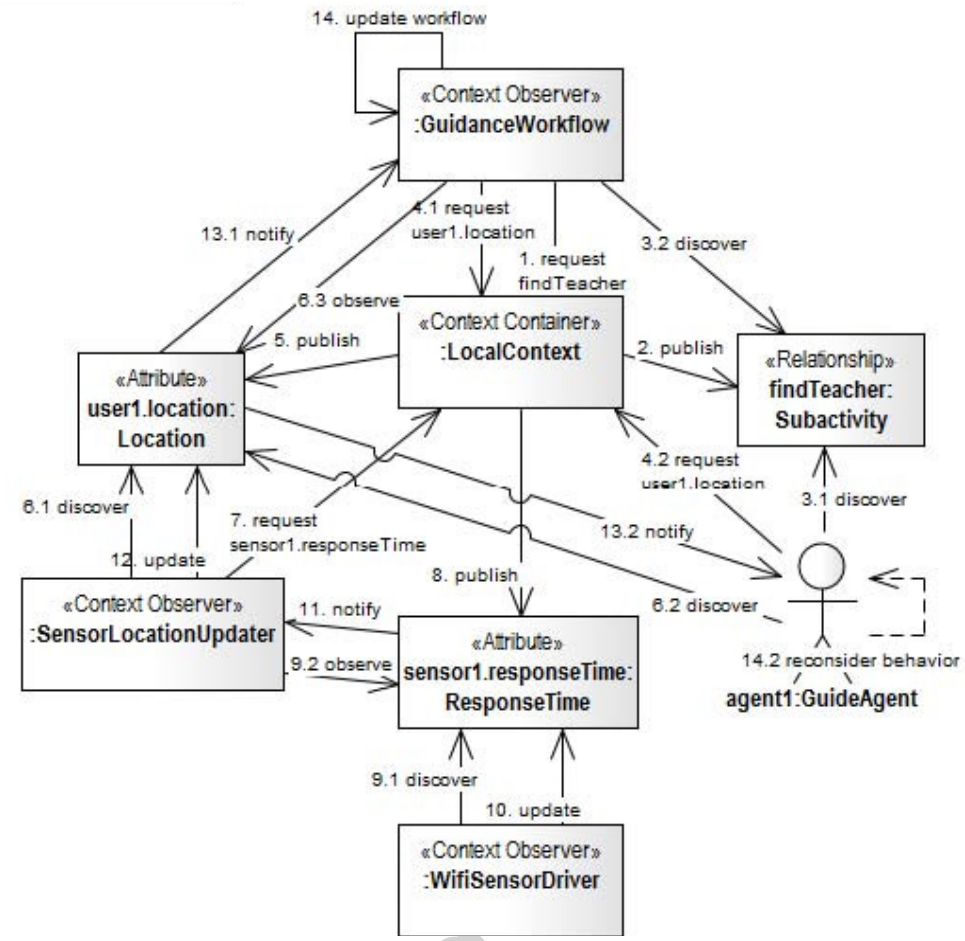

Figure 4. Collaboration Diagram for the Workflow Pattern in the Guidance System.

Figure 4 represents the runtime behavior of the components that implement the *GuidanceWorkflow* from Figure 3. It includes the initialization and evaluation of the context elements needed to update the progress information on the workflow.

The steps involved in the execution of the workflow are:

1. The *GuidanceWorkflow* starts and updates the context to indicate that its current sub-activity is “findTeacher”.

2. The *Context Container* publishes the change.

3. The *GuideAgent* discovers that there is a sub-activity taking place. This causes the agent to create a goal to assist the user in finding the teacher, as described previously.

4. The *GuidanceWorkflow* component requests the *user1.location* element to determine the user location. It needs that information to calculate if the

"findTeacher" sub-activity has already finished. The *GuideAgent* also requests the *user1.location* element. It needs this information to evaluate its objective.

5. The *LocalContext* publishes the request.

6. The *GuidanceWorkflow* and the *GuideAgent* start to observe the element, and a *SensorLocationUpdater* discovers it.

7. The *SensorLocationUpdater* is able to triangulate the user's position from the response time of wifi sensors to the signal from the user's smartphone. Thus, it needs to discover the *sensor1.responseTime* to calculate the position, and it requests this from the *LocalContext*.

8. Subsequently the framework co-ordinates the different components (i.e., the *WifiSensorDriver*, *SensorLocationUpdater*, and *GuidanceWorkflow*) to update the context according to the information provided by the sensors. The same is done with any other information that the agent may need from the environment. In this case, this could be the user voice inputs, and their language and communication preferences.

In the proposed process, once the bindings have been established, the context representation is changed dynamically by the system components to reflect the progress of the situation. Under these conditions, the agent develops its dialog management in complete abstraction from the actual co-ordination or information fusion mechanisms that are being used to provide that information. The lower layers hide their specific details to the upper layers. For instance, as the user walks through the room, the sensors produce a good deal of information. The *SensorLocationUpdater* processes this information, but it updates the *user1.location* element only when the sensor context reflects a change of position. The same is done at the activity level, as the workflow component determines the end of an activity and starts a new one only when the position reaches a given place.

Other available case studies that evaluate the FAERIE infrastructure are an interactive art installation where the viewer moves through different rooms, and different Ambient Assisted Living (AAL) systems.

6. Results

The presented architecture has several important advantages for the development of AmI systems regarding to its modularity and flexibility. Each deployment unit (i.e., *environment*) can work in a completely autonomous way. This allows a quick prototyping of applications providing dummy components for testing if needed.

Moreover, the style of interactions between components does not impose any a prior organization of them to process the context.

The work also introduces a set of mechanisms to carry out the adaptation to the context. This adaptation is performed by defining workflows or context change management policies using the following elements:

- Components to describe the conceptual framework of the application in terms of concepts such as the *Context Sensitive System*, the *environment*, the *context*, the *context container*, the *context observer*, *entities*, *relationships*, and *properties*.
- The already mentioned infrastructure to manage the processing of context.
- A set of control patterns in order to implement the adaptation of systems to context changes.

Engineers have available guidelines for the application of the framework. These describe the tasks to develop context-aware systems following this approach, as well as the aspects to take into account for the different decisions.

7. Conclusions

This paper has presented a way to integrate MAS into a generic architecture for context-awareness and activity management in AmI systems. Its definition includes patterns for developing components that create, manage and use elements of information in the context at different levels of abstraction. Based on these components, it also proposes patterns to develop sub-systems that are able to track context conditions and to perform different actions following activity diagrams.

The integration of MAS and workflow management facilities provides advantages for both types of systems by increasing the abstraction level of behavior definitions. The specification of software agents uses concepts such as goals, protocols and interactions, which are useful for the declarative description of workflows and their relation with actors' purposes. This facilitates in AmI systems the definition of more complex workflows and automated reasoning on them. In addition, the existence of a common "mental state" for agent organizations using the AmI context reduces the need for informative interactions among agents. The main advantage over previous approaches is that the present method uses agents to participate in workflows in a transparent and generic way, i.e., the agents do not need to know either how the conditions are resolved or the concrete workflow in which they are participating.

There are still several open issues in this work. First, there is need to complete the FAERIE architecture and infrastructure for AmI applications with MAS. Given the usual requirements of AmI systems, the architecture will include a *Security*

Management pattern in order to guarantee correct access to sensitive information in context containers. Furthermore, the use of agents to provide complex and emergent interactions in AmI has to be explored. For instance, a future case study would consider several students trying to meet with a teacher at the same time. This conflict would be solved through negotiation among the agents of the students and the teacher, in order to make collective arrangements for the corresponding meetings. This collaboration will use interaction protocols, and could use the shared contexts described in this paper.

Acknowledgments

This work has been done in the context of the project “Social Ambient Assisting Living - Methods (SociAAL)" (TIN2011-28335-C02-01) supported by the Spanish Ministry for Economy and Competitiveness.

References

1. Aiello, F., Bellifemine, F.L., Fortino, G., Galzarano, S., Gravina, R.: An Agent-Based Signal Processing In-Node Environment for Real-Time Human Activity Monitoring Based on Wireless Body Sensor Networks. Engineering Applications of Artificial Intelligence 24(7), 1147-1161 (2011)
2. Aiello, F., Fortino, G., Gravina, R., Guerrieri, A.: A Java-Based Agent Platform for Programming Wireless Sensor Networks. Computer Journal 54(3), 439-454 (2011)
3. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: Context-Aware Workflow Management. In: Baresi, L., Fraternali, P., Houben, G.-J. (eds.) Web Engineering, LNCS, vol. 4607, pp. 47-52. Springer, Heidelberg (2007)
4. Aztiria, A., Basagoiti, R., Augusto, J. C., Espejo, G.: A System to Learn Frequent Behavioural Patterns. In: 8th International Conference on Intelligent Environments (IE 2012), pp. 363-366 (2012)
5. Bergmann, R.: Ambient Intelligence for Decision Making in Fire Service Organizations. In: Schiele, B., Dey, A.K., Gellersen, H., Ruyter, B., Tscheligi, M., Wichert, R., Aarts, E., Buchmann, A. (eds.) Ambient Intelligence, LNCS, vol. 4794, pp. 73-90. Springer, Heidelberg (2007)
6. Blázquez Gil, G., Berlanga, A., Molina, J. M.: InContexto - Multisensor Architecture to Obtain People Context from Smartphones. International Journal of Distributed Sensor Networks, 758789 (2012)
7. Castillo, J.C., Carneiro, D., Serrano-Cuerda, J., Novais, P., Fernández-Caballero, A., Neves, J.: A Multi-Modal Approach for Activity Classification and Fall Detection. International Journal of Systems Science 45(4), 810-824 (2014)
8. Cook, D. J., Augusto, J. C., Jakkula, V. R.: Review - Ambient intelligence: Technologies, applications, and opportunities. Journal Pervasive and Mobile Computing 5(4), 277-298 (2009)

9. Corkill, D.: Blackboard Systems. AI Expert 6(9), 40-47 (1991)
10. Fernández-de-Alba, J., Campillo, P., Fuentes-Fernández, R., Pavón, J.: Opportunistic Sensor Interpretation in a Virtual Smart Environment. In: Yin, H., Costa, J.A.F., Barreto, G. (eds.) Intelligent Data Engineering and Automated Learning – IDEAL 2012, LNCS, vol. 7435, pp. 109-116. Springer, Heidelberg (2012)
11. Fernández-de-Alba, J.M., Fuentes-Fernández, R., Pavón, J.: Dynamic Workflow Management for Context-Aware Systems. In: Novais, P., Hallenborg, K., Tapia, D.I., Corchado-Rodríguez, J.M. (eds.) Ambient Intelligence - Software and Applications, Advances in Intelligent and Soft Computing, vol. 153, pp. 181-188. Springer, Heidelberg (2012)
12. Gómez-Romero, J., Serrano, M. A., Patricio, M. A., García, J., Molina, J. M.: Context-based Scene Recognition from Visual Data in Smart Homes: An Information Fusion approach. Personal and Ubiquitous Computing 16(7), 835-857 (2012)
13. Gómez-Sanz, J.J., Fernández, C.R., Arroyo, J.: Model Driven Development and Simulations with the INGENIAS Agent Framework. Simulation Modelling Practice and Theory 18(10), 1468-1482 (2010)
14. GRASIA Research Group: FAERIE Project (2012) Available at: http://grasia.fdi.ucm.es/faerie/ (last visit: 9 oct. 2014)
15. Guivarch, V., Camps, V., Péninou, A., Stuker, S.: Dynamic Filtering of Useless Data in an Adaptive Multi-Agent System – Evaluation in the Ambient Domain. In: Demazeau, Y., Ishida, T., Corchado, J.M., Bajo, J. (eds.) Advances on Practical Applications of Agents and Multi-Agent Systems – PAAMS 2013, LNCS, vol. 7879, pp. 110-121. Springer, Heidelberg (2013)
16. Han, J., Cho, Y., Kim, E., Choi, J.: A Ubiquitous Workflow Service Framework. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) Computational Science and Its Applications – ICCSA 2006, LNCS, vol. 3983, pp. 30-39. Springer, Heidelberg (2006)
17. Isaza, G., Castillo, A., López, M., Castillo, L., López, M.: Intrusion Correlation Using Ontologies and Multi-agent Systems. In: Bandyopadhyay, S.K., Adi, W., Kim, T.-H., Xiao, Y. (eds.) Information Security and Assurance – ISA 2010, Communications in Computer and Information Science, vol. 76, pp. 51-63. Springer, Heidelberg (2010)
18. Kopp, S., Gesellensetter, L., Krämer, N.C., Wachsmuth, I.: A Conversational Agent as Museum Guide – Design and Evaluation of a Real-World Application. In: Panayiotopoulos, T., Gratch, J., Aylett, R., Ballin, D., Olivier, P., Rist, T. (eds.) Intelligent Virtual Agents, LNCS, vol. 3661, pp. 329-343. Springer, Heidelberg (2005)
19. Martí, E. D., Martín, D., García, J., de la Escalera, A., Molina, J. M., Armingol, J. M.: Context-Aided Sensor Fusion for Enhanced Urban Navigation. Sensors 12(12), 16802-16837 (2012)
20. Novais, P., Carneiro, D., Gomes, M., Neves, J.: Non-invasive Estimation of Stress in Conflict Resolution Environments. In : Advances on Practical Applications of Agents and Multi-Agent Systems - 10th International Conference on Practical Applications of Agents and Multi-Agent Systems,

Advances in Intelligent and Soft Computing, vol. 155, pp. 153-160. Springer, Heidelberg (2012)
21. Preuveneers, D., Novais, P.: A survey of software engineering best practices for the development of smart applications in Ambient Intelligence. Journal of Ambient Intelligence and Smart Environments 4(3), 149-162 (2012)
22. Ramos, C., Augusto, J.C., Shapiro, D.: Ambient Intelligence - The Next Step for Artificial Intelligence. IEEE Intelligent Systems 23(2), 15-18 (2008)
23. Ranganathan, A., McFaddin, S.: Using Workflows to Coordinate Web Services in Pervasive Computing Environments. In: 2004 IEEE International Conference on Web Services (ICWS 2004). pp. 288-295. IEEE Computer Society (2004)
24. Sánchez-Pi, N., Molina, J. M.: A Multi-Agent Approach for Provisioning of e-Services in u-Commerce Environments. Internet Research 20(3), 276-295 (2010)
25. Sánchez,-Pi, N., Carbó, J., Molina, J. M.: A knowledge-based system approach for a context-aware system. Knowledge-Based Systems 27, 1-17 (2012)
26. Silva, F., Analide, C., Novais, P.: Information Fusion for Context Awareness in Intelligent Environments. In: Pan, J.-S., Polycarpou, M.M., Woźniak, M., de Carvalho, A.C.P.L.F., Quintián, H., Corchado, E. (eds.) Hybrid Artificial Intelligent Systems – HAIS 2013, LNCS, vol. 8073, pp. 252-261. Springer, Heidelberg (2013)
27. Tapia, D.I., Abraham, A., Corchado, J.M., Alonso, R.S.: Agents and ambient intelligence: case studies. Journal of Ambient Intelligence and Humanized Computing 1(2), 85-93 (2010)
28. Tapia, D.I., Fraile, J.A., Rodríguez, S., Alonso, R.S., Corchado, J.M.: Integrating Hardware Agents into an Enhanced Multi-Agent Architecture for Ambient Intelligence Systems. Information Sciences 222, 47-65 (2013)
29. Venturini, V., Carbó, J., Molina, J.M.: Methodological Design and Comparative Evaluation of a MAS Providing AmI. Expert Systems with Applications 39(12), 10656-10673 (2012)

Implementation of Context-Aware Workflows with Multi-Agent Systems

Javier Alfonso-Cendón[1], José M. Fernández-de-Alba[2], Rubén Fuentes-Fernández[2], and Juan Pavón[2]
[1] Escuela de Ingenierías Industrial e Informática, Universidad de León
24071 León, Spain
javier.alfonso@unileon.es
[2]Facultad de Informática de la Universidad Complutense de Madrid
Avda. Complutense, s/n. 28040 Madrid (Spain)
{jmfernandezdealba,ruben,jpavon}@fdi.ucm.es

Short biography of authors

Javier Alfonso Cendón is PhD in Engineering by the University of Leon, Master MBA by the Polytechnic University of Madrid, Master in Renewable Energies by the University of Leon, Master in Security by University of Leon / INTECO, and Computer Engineer by the University of León. He has many specialized courses in the field of engineering and communication technologies. He is PhD Assistant Professor in the area of engineering projects in the University of León. He has more than 20 national and international publications in the field of ICT.

José M. Fernández de Alba is currently a PhD candidate at Universidad Complutense Madrid. He has been working on distributed multimodal dialog systems and ambient intelligence. Recently he has developed a context-aware architecture, FAERIE, which is based on components and methodology making use of concepts like test-driven development and continuous integration.

Rubén Fuentes is Associate Professor at Universidad Complutense Madrid and member of the UCM-GRASIA Research Group. He got a PhD degree in Computer Science from Universidad Complutense Madrid (2004). He has published more than 50 research papers in international conferences and journals. Previously, he worked as a consultant in database systems for four years. His main interests are concerned with the application of Social Sciences to software development, model-driven engineering, agent-oriented methodologies, and social simulation.

Juan Pavón is Full Professor at Universidad Complutense Madrid. He got a PhD degree in Computer Science from Universidad Politécnica Madrid (1988). From 1987 to 1997 he was working in R&D departments of Alcatel in the development of component-based architectures for distributed systems and their application to multimedia services on broadband networks and mobile systems. He joined UCM at the end of 1997, where he created the UCM-GRASIA research group, whose focus is on the application of multi-agent systems technology, in particular, on software engineering, knowledge management, simulation of complex systems, decision making, interactive art, and ambient assisted living.

*Photo of the author(s)

*Photo of the author(s)

*Photo of the author(s)

*Photo of the author(s)